Constraint Programming

Prof. Dr. Thom Frühwirth

Faculty of Computer Science
University of Ulm, Germany
www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/

October 2005

Citations may be incomplete for space reasons

Constraint Reasoning and Programming

Part I

Constraint Programming

The Holy Grail

Constraint Programming represents one of the closest approaches computer science has yet made to the **Holy Grail** of programming: the user states the problem, the computer solves it.

Eugene C. Freuder, Inaugural issue of the *Constraints Journal*, 1997.

Constraint Reasoning

The Idea

- *Combination Lock Example*
  0 1 2 3 4 5 6 7 8 9
  Greater or equal 5.
  Prime number.
- *Declarative problem* representation by variables and constraints:
  $x \in \{0, 1, \ldots, 9\} \ \wedge \ x \geq 5 \ \wedge \ \text{prime}(x)$
- *Constraint propagation and simplification* reduce search space:
  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \ \rightarrow \ x \in \{5, 6, 7, 8, 9\}$

Constraint Reasoning Everywhere

Combination

Simplification

Contradiction

Redundancy

Terminology

Language is first-order logic with equality.

- Constraint:
  Conjunction of atomic constraints (predicates)
  E.g., $4X + 3Y = 10 \ \wedge \ 2X - Y = 0$
- Constraint Problem (Query):
  A given, initial constraint
- Constraint Solution (Answer):
  A valuation for the variables in a given constraint problem that satisfies all constraints of the problem. E.g., $X = 1 \wedge Y = 2$.

  In general, a normal/solved form of the constraints.
  E.g., the problem $4X + 3Y + Z = 10 \ \wedge \ 2X - Y = 0$
  simplifies into $Y + Z = 10 \ \wedge \ 2X - Y = 0$.

Mortgage

D: Amount of Loan, Debt, Principal
T: Duration of loan in months
I: Interest rate per month
R: Rate of payments per month
S: Balance of debt after T months

```
mortgage(D, T, I, R, S) <=>
       T = 0,
       D = S
         ;
       T > 0,
       T1 = T - 1,
       D1 = D + D*I - R,
       mortgage(D1, T1, I, R, S).
```

Mortgage II

```
mortgage(D, T, I, R, S) <=>
        T = 0, D = S
          ;
        T > 0, T1 = T - 1, D1 = D + D*I - R,
        mortgage(D1, T1, I, R, S).
```

- `mortgage(100000,360,0.01,1025,S)` yields `S=12625.90`.
- `mortgage(D,360,0.01,1025,0)` yields `D=99648.79`.
- `mortgage(100000,T,0.01,1025,S), S=<0` yields `T=374, S=-807.96`.
- `mortgage(D,360,0.01,R,0)` yields `R=0.0102861198*D`.

Advantages of Constraint Logic Programming

Theoretical
Logical Foundation – First-Order Logic

Conceptual
Sound Modeling

Practical
Efficient Algorithms/Implementations
Combination of different Solvers

Constraint Reasoning and Programming

Generic Framework for

- Modeling
  - with partial information
  - with infinite information
- Reasoning
  - with new information
- Solving
  - combinatorial problems

The Appeal of Constraint Programming

Robust, flexible, maintainable software faster.

- Declarative modeling by constraints:
  Description of properties and relationships between partially known objects.
  Correct handling of precise and imprecise, finite and infinite, partial and full information.
- Automatic constraint reasoning:
  Propagation of the effects of new information (as constraints).
  Simplification makes implicit information explicit.
- Solving combinatorial problems efficiently:
  Easy Combination of constraint solving with search and optimization.

Early Commercial Applications (in the 90s)

- Lufthansa: Short-term staff planning.
- Hongkong Container Harbor: Resource planning.
- Renault: Short-term production planning.
- Nokia: Software configuration for mobile phones.
- Airbus: Cabin layout.
- Siemens: Circuit verification.
- Caisse d'epargne: Portfolio management.

In Decision Support Systems for **Planning and Configuration**, for **Design and Analysis**.

Early History of Constraint Programming

60s Constraint networks in artificial intelligence.
70s Logic programming (Prolog).
80s Constraint logic programming.
80s Concurrent logic programming.
90s Concurrent constraint programming.
90s Commercial applications.

Constraint Reasoning Algorithms

Adaption and combination of existing efficient algorithms from

- Mathematics
  - Operations research
  - Graph theory
  - Algebra
- Computer Science
  - Finite automata
  - Theorem proving
- Economics
- Linguistics

Application Domains

- Modeling
- Executable Specifications
- Solving Combinatorial Problems

Scheduling, Planning, Timetabling
Configuration, Layout, Placement, Design
Analysis: Simulation, Verification, Diagnosis
of **software, hardware and industrial processes**.

Applications in Research

- Artificial Intelligence
  - Machine Vision
  - Natural Language Understanding
  - Temporal and Spatial Reasoning
  - Theorem Proving
  - Qualitative Reasoning
  - Robotics
  - Agents

Applications in Research II

- Computer Science: Program Analysis, Robotics, Agents
- Molecular Biology, Biochemestry, Bioinformatics:
  Protein Folding, Genomic Sequencing
- Economics: Scheduling
- Linguistics: Parsing
- Medicine: Decision Support
- Physics: System Modeling
- Geography: Geo-Information-Systems

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
      [S,E,N,D,M,O,R,Y] in 0..9,
      S≠0, M ≠0,
      alldifferent([S,E,N,D,M,O,R,Y]),
                  1000*S + 100*E + 10*N + D
      +           1000*M + 100*O + 10*R + E
      = 10000*M + 1000*O + 100*N + 10*E + Y,
      labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
      [S,E,N,D,M,O,R,Y] in 0..9,
      S≠0, M ≠0,
      alldifferent([S,E,N,D,M,O,R,Y]),
                  1000*S + 100*E + 10*N + D
      +           1000*M + 100*O + 10*R + E
      = 10000*M + 1000*O + 100*N + 10*E + Y,
      labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
= & M & O & N & E & Y
\end{array}
$$

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

Crypto-Arithmetic Problem

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
= & M & O & N & E & Y
\end{array}
$$

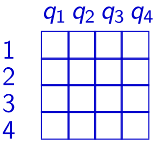
```
solve(S,E,N,D,M,O,R,Y) :-
         [S,E,N,D,M,O,R,Y] in 0..9,
        S≠0, M ≠0,
        alldifferent([S,E,N,D,M,O,R,Y]),
                    1000*S + 100*E + 10*N + D
        +           1000*M + 100*O + 10*R + E
        = 10000*M + 1000*O + 100*N + 10*E + Y,
        labeling([S,E,N,D,M,O,R,Y]).
```

S=9, E in 4..7, N in 5..8, M=1, O=0, [D,R,Y] in 2..8

With Search: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

n-Queens Problem

Place n queens $q_1, \ldots, q_n$ on an $n \times n$ chess board, such that they do not attack each other.



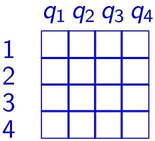
$$q_1, \ldots, q_n \in \{1, \ldots, n\}$$

$$\forall\ i \neq j.\ q_i \neq q_j \wedge |q_i - q_j| \neq |i - j|$$

- no two queens on same row, column or diagonal
  - each row and each column with exactly one queen
  - each diagonal at most one queen
- q_i: row position of the queen in the i-th column

n-Queens Problem II

Place n queens $q_1, \ldots, q_n$ on an $n \times n$ chess board,
such that they do not attack each other.



$$q_1, \ldots, q_n \in \{1, \ldots, n\}$$

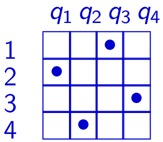
$$\forall\ i \neq j.\ q_i \neq q_j \wedge |q_i - q_j| \neq |i - j|$$

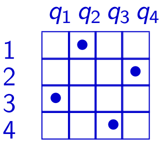
```
solve(N,Qs)         <=> makedomains(N,Qs), queens(Qs), enum(Qs).
queens([Q|Qs])      <=> safe(Q,Qs,1), queens(Qs).
safe(X,[Y|Qs],N)    <=> noattack(X,Y,N), safe(X,Qs,N+1).
noattack(X,Y,N)     <=> X ne Y, X+N ne Y, Y+N ne X.
```

n-Queens Problem III

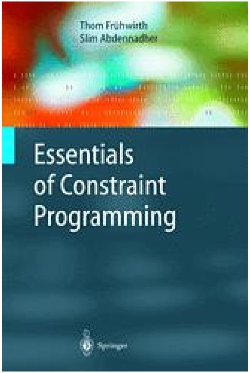
`solve(4,[Q1,Q2,Q3,Q4])`

- `makedomains` produces possible positions for queens
  `Q1 in [1,2,3,4], Q2 in [1,2,3,4]`
  `Q3 in [1,2,3,4], Q4 in [1,2,3,4]`
- `safe` adds `noattack` for each ordered pair of queens
- `noattack` produces `ne` constraints between queens
- `enum` called for labeling using the domains of queens
- `[Q1,Q2,Q3,Q4] = [2,4,1,3]`, `[Q1,Q2,Q3,Q4] = [3,1,4,2]`





Further Reading



Essentials of Constraint Programming
Thom Frühwirth,
Slim Abdennadher
Springer, 2003.

Constraint-Programmierung
Lehrbuch
Thom Frühwirth,
Slim Abdennadher
Springer, 1997.

Overview

- Basic First-Order Logic
- Constraint Programming Languages
    - Constraint logic programming (Prolog, CLP)
    - Concurrent committed-choice constraint logic programming (CC)
    - Constraint handling rules (CHR)

  For each language:
    - Syntax
    - Declarative and Operational Semantics
    - Soundness and Completeness
- Constraint Systems
    - Rational Trees, Feature Terms, Description Logic
    - Boolean Constraints
    - Finite and Interval Domains
    - Linear and Non-Linear Polynomial Equations

  For each system:
    - Constraint Theory
    - Solving Algorithms
    - Applications